

Stochastic Runtime Analysis of a Cross-Entropy Algorithm for Traveling Salesman Problems

Zijun Wu^{a,1,*}, Rolf H. Möhring^{b,2,**}, Jianhui Lai^c,

^a*Beijing Institute for Scientific and Engineering Computing (BISec), Pingle Yuan 100, Beijing, P. R. China*

^b*Beijing Institute for Scientific and Engineering Computing (BISec), Pingle Yuan 100, Beijing, P. R. China*

^c*College of Metropolitan Transportation, Beijing University of Technology, Pingle Yuan 100, Beijing, P. R. China*

Abstract

This article analyzes the stochastic runtime of a Cross-Entropy Algorithm on two classes of traveling salesman problems. The algorithm shares main features of the famous Max-Min Ant System with iteration-best reinforcement.

For simple instances that have a $\{1, n\}$ -valued distance function and a unique optimal solution, we prove a stochastic runtime of $O(n^{6+\epsilon})$ with the vertex-based random solution generation, and a stochastic runtime of $O(n^{3+\epsilon} \ln n)$ with the edge-based random solution generation for an arbitrary $\epsilon \in (0, 1)$. These runtimes are very close to the known expected runtime for variants of Max-Min Ant System with best-so-far reinforcement. They are obtained for the stronger notion of stochastic runtime, which means that an optimal solution is obtained in that time with an overwhelming probability, i.e., a probability tending exponentially fast to one with growing problem size.

We also inspect more complex instances with n vertices positioned on

*Principal corresponding author

**Corresponding author

Corresponding author

Email addresses: `zijunwu@bjut.edu.cn` (Zijun Wu), `rolf.moehring@tu-berlin.de` (Rolf H. Möhring), `laijianhui@bjut.edu.cn` (Jianhui Lai)

¹The author is also affiliated with the School of Applied Mathematics and Physics at Beijing University of Technology

²The author is a professor emeritus of mathematics at Berlin University of Technology

an $m \times m$ grid. When the n vertices span a convex polygon, we obtain a stochastic runtime of $O(n^3 m^{5+\epsilon})$ with the vertex-based random solution generation, and a stochastic runtime of $O(n^2 m^{5+\epsilon})$ for the edge-based random solution generation. When there are $k = O(1)$ many vertices inside a convex polygon spanned by the other $n - k$ vertices, we obtain a stochastic runtime of $O(n^4 m^{5+\epsilon} + n^{6k-1} m^\epsilon)$ with the vertex-based random solution generation, and a stochastic runtime of $O(n^3 m^{5+\epsilon} + n^{3k} m^\epsilon)$ with the edge-based random solution generation. These runtimes are better than the expected runtime for the so-called $(\mu + \lambda)$ EA reported in a recent article, and again obtained for the stronger notion of stochastic runtime.

Keywords: probabilistic analysis of algorithms, stochastic runtime analysis of evolutionary algorithms, Cross Entropy algorithm, Max-Min Ant System, $(\mu + \lambda)$ EA.

1. Introduction

The Cross Entropy (CE) algorithm is a general-purpose evolutionary algorithm (EA) that has been applied successfully to many \mathcal{NP} -hard combinatorial optimization problems, see e.g. the book [1] for an overview. It was initially designed for rare event simulation by Rubinstein [2] in 1997, and thereafter formulated as an optimization tool for both continuous and discrete optimization (see [3]).

CE has much in common with the famous ant colony optimization (ACO, see [4]) and the estimation of distribution algorithms (EDAs, see [5]). They all belong to the so-called *model-based search* paradigm (MBS), see [6]. Instead of only manipulating solutions, which is very typical in traditional heuristics like Genetic Algorithms [7] and Local Search [8] and others, MBS algorithms attempt to optimize the solution reproducing mechanism. In each iteration, they produce new solutions by sampling from a probabilistic distribution on the search space. The distribution is often called a *model* in the literature (see e.g. [6] and [9]). This model evolves iteratively by incorporating information from some elite solutions occurring in the search history, so as to asymptotically model the spread of optimal solutions in the search space. See the recent Thesis [9] for more details on MBS algorithms and their mathematical properties.

An important issue for these MBS algorithms is to determine a suitable size for the sampling in each iteration. A large sample size obviously makes

each iteration unwieldy, however a small sample size may mislead the underlying search due to the randomness in the sampling. Sample size actually reflects the *iterative complexity* (computational complexity in each iteration) of an MBS algorithm. However, whether a large sample size is harmful depends on the required *iteration number* (i.e., the total number of iterations required to reach an optimal solution). This article aims to shed a light on this issue by theoretically analyzing the relation between sample size and iteration number for a CE *variant* that includes also some essential features of the famous Max-Min Ant System (*MMAS* [10]). To this end, a thorough runtime analysis is needed.

The theoretical runtime analysis of evolutionary algorithms has gained rapidly growing interest in recent years, see e.g. [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], and [23]. In these papers, an oracle-based view of computation is adopted, i.e., the *runtime* of an algorithm is expressed as *the total number of solutions evaluated before reaching an optimal solution*. Since the presence of randomness, the runtime of an EA is often conveyed in expectation or with high probability. Due to the famous No Free Lunch Theorem [24], runtime analysis must be problem-specific. The first steps towards this type of runtime analysis were made for the simple evolutionary algorithm (1+1) EA [11] on some test problems that use *pseudo boolean functions* as cost functions, e.g., ONEMAX [15], LEADINGONES [25] and BINVAR [11]. Recent research addresses more and more problems of practical importance, such as the computing a minimum spanning trees (MST) [26], matroid optimization [27], traveling salesman problem [28], the shortest path problem [22], the maximum satisfiability problem [29] and the max-cut problem [30].

Runtime analysis in the literature considers two cases: *expected runtime analysis* and *stochastic runtime analysis*. Expected runtime is the average runtime of an algorithm on a particular problem, see, e.g., the runtime results of (1 + 1) EA reported in [11]. Expected runtime reflects the oracle-based average performance of an algorithm. A mature technique for expected runtime analysis is the so-called drift analysis [12]. However, this technique requires that the algorithm has a finite expected runtime for the underlying problem. Due to the stagnation behavior reported in [31], drift analysis is not applicable to the *traditional* CE [3].

Note that an algorithm with a smaller expected runtime need not be more efficient, see [32] for details. In contrast, stochastic runtime can provide a better understanding of the performance of a (randomized) EA. Stochas-

tic runtime is a runtime result conveyed with an overwhelming probability guarantee (see, e.g., the classic runtime result of 1-ANT in [15]), where an overwhelming probability means a probability that tends to 1 exponentially fast in the problem size. It therefore reflects the efficiency of an algorithm for most cases in the sense of uncertainty. This article is concerned with stochastic runtime analysis. Our analysis shall closely inspect the probability of the random event that the runtime is bounded from above by a polynomial in the problem size, and investigate the relation between the polynomial upper bound and the sample size.

The runtime analysis of CE algorithms has been initiated in [31], where Wu and Kolonko proved a pioneering stochastic runtime result for the traditional CE on the standard test problem LEADINGONES. As a continuation of the study of [31], Wu et al [32] further investigated the stochastic runtime of the traditional CE on another test problem ONEMAX. The runtime results reported in [31] and [32] showed that sample size plays a crucial role in efficiently finding an optimal solution. In particular, Wu et al [32] showed that if the sample size is moderately adapted to the problem size n , then the stochastic runtime of the traditional CE on ONEMAX is $O(n^{1.5+\frac{4}{3}\epsilon})$ for arbitrarily small $\epsilon > 0$ and a constant smoothing parameter $\rho > 0$, which beats the best-known stochastic runtime $O(n^2)$ reported in [13] for the classic 1-ANT algorithm, although 1-ANT employs a much smaller sample size (i.e., sample size equals one). Moreover, by imposing upper and lower bounds on the sampling probabilities as was done in *MMAS* [10], Wu et al [32] showed further that the stochastic runtime of the resulting CE can be significantly improved even in a very rugged search space.

The present article continues the stochastic runtime analysis of [32], but now in combinatorial optimization with a study of CE on the traveling salesman problem (TSP). We consider a CE variant that resembles a *MMAS* with iteration-best reinforcement under two different random solution generation mechanisms, namely, a vertex-based random solution generation and an edge-based random solution generation.

TSP is a famous \mathcal{NP} -complete combinatorial optimization problem. It concerns finding a shortest Hamiltonian cycle on a weighted complete graph. Existing algorithms exactly solving TSP generally have a prohibitive complexity. For instance, the Held-Karp algorithm [33] solves the problem with a complexity of $O(n^2 2^n)$. A well-known polynomial time approximation algorithm for metric TSP is the so-called Christofides algorithm [34], which finds a solution with a cost at most $3/2$ times the cost of optimal solutions.

As mentioned in [35], this is still the best known approximation algorithm for the general metric TSP so far. For Euclidean TSP there exists a famous polynomial-time approximation scheme (PTAS) by Arora, see [36]. To design a superior approximation algorithm, researchers in recent years tend to study TSP instances with particular structures, see, e.g., [37].

Due to the prohibitive running time of exact algorithms, heuristics are frequently employed in practice so as to efficiently compute an acceptable solution for a TSP problem, e.g., the Lin-Kernighan (LK) algorithm [38]. As a popular heuristic, CE has also been applied in practice to solve TSP instances, see [39] and [3]. The implementation there shows that CE can also efficiently compute an acceptable solution.

In view of the high complexity of general TSP, we consider in our analysis two classes of TSP instances with a particular structure. The first kind of instances has been used in [19] and [40] for analyzing the expected runtime of some *MMAS* variants. These TSP instances have polynomially many objective function values and a unique optimal solution, and so solutions containing more edges from the optimal solution have a lower cost than those with fewer such edges. For more details on these instances, see Section 4.

For these simple instances, we show a stochastic runtime of $O(n^{6+\epsilon})$ for the vertex-based random solution generation, and a stochastic runtime of $O(n^{3+\epsilon} \ln n)$ for the edge-based random solution generation, for any constant $\epsilon \in (0, 1)$ and a sample size $N = n^\epsilon$, see Theorem 2 for details. These results are very similar to the known expected runtime $O(n^6 + \frac{n \ln n}{\rho})$ for $(1+1)$ MMAA reported in [19], and the expected runtime $O(n^3 \ln n + \frac{n \ln n}{\rho})$ for *MMAS*_{Arb}^{*} reported in [40], where $\rho \in (0, 1)$ is an evaporation rate. But they give the stronger guarantee of achieving the optimal solution in the respective runtime with an overwhelming probability.

We also inspect more complex instances with n vertices positioned on an $m \times m$ grid, and the Euclidean distance as distance function. These instances have been employed in [41] and [28] for analyzing the expected runtime of $(\mu + \lambda)$ EA and randomized local search (RLS). When the n vertices span a convex polygon without vertices in the interior of the polygon (so they are the corners of that polygon), we prove a stochastic runtime of $O(n^3 m^{5+\epsilon})$ for the vertex-based random solution generation, and a stochastic runtime of $O(n^2 m^{5+\epsilon})$ for the edge-based random solution generation, see Theorem 3 for details. When the vertices span a convex polygon with $k = O(1)$ vertices in the interior, we show a stochastic runtime of $O(n^4 m^{5+\epsilon} + n^{6k-1} m^\epsilon)$ with

the vertex-based random solution generation, and a stochastic runtime of $O(n^3 m^{5+\epsilon} + n^{3k} m^\epsilon)$ with the edge-based random solution generation, see Theorem 4 for details. These runtimes are better than the expected runtime for the so-called $(\mu + \lambda)$ EA and RLS reported in the recent paper [28].

The remainder of this paper is arranged as follows. Section 2 defines the traditional CE and related algorithms, Section 3 defines the traveling salesman problem and provides more details of the used CE variants, and Section 4 reports the stochastic runtime results on the TSP instances. A short conclusion and suggestions for future work are given in Section 5.

Notations for runtime

Our analysis employs some commonly used notations from complexity theory. We use $O(f(n))$ to denote the class of functions which are *bounded above* by the function $f(n)$, i.e., those functions $g(n)$ with $g(n) \leq c \cdot f(n)$ for some constant $c \geq 0$ not depending on n . Similarly, $\Omega(f(n))$ is the class of functions that are *bounded below* by $f(n)$, i.e., for any $g(n) \in \Omega(f(n))$ there exists a constant $c > 0$ not depending on n such that $g(n) \geq c \cdot f(n)$. Class $\Theta(f(n))$ is the intersection of $\Omega(f(n))$ and $O(f(n))$. Class $o(f(n))$ is the class of functions $g(n)$ with $g(n)/f(n) \rightarrow 0$ as $n \rightarrow \infty$, and class $\omega(f(n))$ is the class of functions $g(n)$ with $g(n)/f(n) \rightarrow +\infty$ as $n \rightarrow \infty$. Obviously, $o(f(n)) \subset O(f(n))$ and $\omega(f(n)) \subset \Omega(f(n))$.

2. The general cross entropy algorithm and related algorithms

We now introduce the traditional CE algorithm. The CE variant we will analyze inherits the framework of this traditional version. To compare our results with those in the literature, we shall give also details about some related algorithms.

2.1. The traditional cross entropy algorithm

Algorithm 1 lists the traditional CE that was proposed in [3], adapted to an abstract notion of combinatorial optimization problems. The algorithm assumes a combinatorial minimization problem (S, f) , where S is a *finite* search space of “feasible” solutions and f is the *cost* function. Every feasible solution $s \in S$ is composed of elements from a fixed finite set \mathcal{A} , the ground set of the problem, i.e., we assume $S \subseteq \mathcal{A}^n$ for some integer $n \in \mathbb{N}$. Furthermore there is a product distribution on the product space \mathcal{A}^n that induces a

distribution on $S \subseteq \mathcal{A}^n$. The distribution on \mathcal{A}^n can usually be represented as a vector (or matrix) of real-valued probabilities. The convex combination of the two distributions in Step 6 of Algorithm 1 then corresponds to a convex combination of the two vectors (or matrices).

Specific to the TSP, the ground set \mathcal{A} can be the set of nodes or edges, n is the number of nodes, and a feasible solution is a sequence of elements from \mathcal{A} that forms a Hamiltonian cycle. The product distribution for the TSP is represented as an $n \times n$ matrix.

When we consider the set of nodes as our ground set \mathcal{A} , each row i of the matrix is a marginal distribution that specifies choice probabilities for *all* nodes following the current node i . A random Hamiltonian cycle is sequentially constructed from the product distribution by allowing only nodes not yet visited as continuations in each step, see Algorithm 2 for more details.

When we consider the set of edges as \mathcal{A} , marginals of the product distribution will be represented by the same $n \times n$ matrix where the sum of the (i, j) -th and (j, i) -th entries reflects the probability that the edge $\{i, j\}$ occurs in a random solution. A random Hamiltonian cycle is still constructed sequentially and only edges leading to a feasible solution are taken in each step, see Algorithm 3 for details.

Algorithm 1 The general Cross-Entropy algorithm

Input:

an *initial distribution* Π_0 on the solution space, a fixed *smoothing parameter* $\rho \in (0, 1]$, a *sample size* $N \in \mathbb{N}_+$, an *elite size* $M \in \mathbb{N}_+$ with $M \leq N$

- 1: $t = 0$;
 - 2: **loop**
 - 3: independently generate N many random solutions $\mathbf{X}_t^{(1)}, \dots, \mathbf{X}_t^{(N)}$ with the current distribution Π_t ;
 - 4: sort these N solutions in non-decreasing order as $f(\mathbf{X}_t^{[1]}) \leq \dots \leq f(\mathbf{X}_t^{[N]})$ according to the cost function f ;
 - 5: learn an empirical distribution \mathbf{W}_t from the M many best solutions $\mathbf{X}_t^{[1]}, \dots, \mathbf{X}_t^{[M]}$;
 - 6: set $\Pi_{t+1} = (1 - \rho)\Pi_t + \rho\mathbf{W}_t$;
 - 7: $t = t + 1$;
 - 8: **end loop**
-

Traditionally, CE sets a small elite ratio $\alpha \in (0, 1)$ and uses the best $\lfloor \alpha \cdot N \rfloor$ many solutions in Step 5 to build the empirical distribution \mathbf{W}_t . Here, we use the elite size M instead. This does not intrinsically change the original algorithm. Steps 3 and 5 depend on the detailed definition of the underlying problem. We shall give details to them in Subsection 3.2.

Step 6 of Algorithm 1 plays a crucial role in the different theoretical analyses of the algorithm, see, e.g., [42], [31], [43], [9], [32]. The occurrence of good solutions are probabilistically enforced by incorporating the new information \mathbf{W}_t into $\mathbf{\Pi}_{t+1}$. This idea, somehow, coincides with the reinforcement learning in [44]. The smoothing parameter ρ reflects the relative importance of the new information \mathbf{W}_t in the next sampling. It balances global exploration and local exploitation to a certain degree. A larger ρ makes the algorithm concentrate more on the particular area spanned by the elite solutions $\mathbf{X}_t^{[1]}, \dots, \mathbf{X}_t^{[M]}$, while a smaller ρ gives more opportunities to solutions outside that area.

However, balancing global exploration and local exploitation through tuning ρ is ultimately limited. Wu and Kolonko [31] proved that the famous “genetic drift” [45] phenomenon also happens in this algorithmic scheme, i.e., the sampling (Step 3) eventually freezes at a single solution and that solution needs not to be optimal. This means that the algorithm gradually loses the power of global exploration.

As a compensation for global exploration, Wu et al [32] proved that a moderately large sample size N might be helpful. The results there showed that a moderately large N configured with a large ρ (e.g., $\rho = 1$) can make the algorithm very efficient. Although a large N introduces a high computational burden in each iteration, the total number of iterations required for getting an optimal solution is considerably reduced.

Wu et al [32] also indicated another way to compensate the global exploration, i.e., imposing a lower bound $\pi_{\min} \in (0, 1)$ and an upper bound $\pi_{\max} \in (0, 1)$ on the sampling distributions in each iteration. This idea is originated from \mathcal{MMAS} [10]. In each iteration t , *after* applying Step 6, the entries of distribution $\mathbf{\Pi}_{t+1}$ that are out of the range $[\pi_{\min}, \pi_{\max}]$ are reset to that range by assigning to them the nearest bounds, see (6) in Section 3 for more details. Wu et al [32] have proved that this can make CE more efficient even in the case of a rugged search space.

To follow these theoretical suggestions made in [32], we shall in our stochastic runtime analysis use a CE that modifies the traditional CE (Algorithm 1) accordingly. We shall see that these modifications make the CE

very efficient for the considered TSP instances.

2.2. Related evolutionary algorithms

Related evolutionary algorithms for TSP whose runtime has been extensively studied are RLS [26], $(\mu + \lambda)$ EA [28], and those theoretical abstractions of \mathcal{MMAS} [10] including \mathcal{MMAS}_{bs}^* [17], (1+1) MMAA [19]. We now give algorithmic details of them. In order to facilitate the comparison, their runtimes for TSP instances will be discussed in Section 4.

$(\mu+\lambda)$ EA is an extension of the famous (1+1) EA [11]. $(\mu+\lambda)$ EA randomly chooses μ many solutions as the initial population. In each iteration, $(\mu+\lambda)$ EA randomly chooses λ parents from current population, then produces λ children by applying randomized mutation to each of the selected parents, and forms the next population by taking the best μ many solutions from these $\mu + \lambda$ many solutions in the end of current iteration. The expected runtime of $(\mu+\lambda)$ EA on TSP instances is studied in [28], where Sutton et al uses a Poisson distribution to determine the number of randomized mutations (2-opt move or jump operation) should be taken by a selected parent in each iteration.

RLS is a local search technique [46]. It employs a randomized neighborhood. In each iteration, it randomly chooses a number of components of the best solution found so far and then changes these components. The expected runtime of RLS for TSP instances is also studied in [28], where the neighborhood is taken to be a k -exchange neighborhood with k randomly determined by a Poisson distribution.

(1+1) MMAA is a simplified version of the famous \mathcal{MMAS} [10], where the sample size is set to 1 and pheromones are updated only with the best solution found so far (*best-so-far reinforcement*) in each iteration. In each iteration of (1+1) MMAA, the ant which constructed the best solution found so far deposits an amount π_{\max} of pheromones on the traversed edges, and an amount π_{\min} of pheromones on the non-traversed edges, and the pheromones are updated by linearly combining the old and these newly added pheromones as in Algorithm 1. The expected runtime of (1+1) MMAA on simple TSP instances is studied in [19]. The expected runtime of its variant \mathcal{MMAS}_{Arb}^* on simple TSP instances is studied in [40].

3. The traveling salesman problem and details of the CE variant

Now, we formally define TSP, and give more details of the CE variant we will analyze.

3.1. The traveling salesman problem

We consider an undirected graph $G = (V, E)$ with vertex set $V = \{1, \dots, n\}$ and edge set $E = \{\{i, j\} \mid i \in V, j \in V, i \neq j\}$. A *Hamiltonian cycle* is a sequence $\{\{i_l, i_{l+1}\} \mid l = 1, \dots, n\}$ of edges such that

- a) $i_1 = i_{n+1}$;
- b) (i_1, \dots, i_n) is a permutation of $\{1, 2, \dots, n\}$.

This definition actually considers E as the ground set \mathcal{A} . As mentioned above, we can also put $\mathcal{A} = V$ and represent Hamiltonian cycles in a more compact way as permutations of V . Note that a Hamiltonian cycle corresponds to n many different permutations, whereas a permutation corresponds to a unique Hamiltonian cycle. However, the two representations are intrinsically the same. We shall use them interchangeably in the sequel. To facilitate our discussion, we shall refer to a Hamiltonian cycle by just referring to one of the n corresponding permutations, and denote by S the set of all possible permutations. We employ the convention that two permutations are said to be same iff they form the same underlying Hamiltonian cycle. The notation $\{k, l\} \in s$ shall mean that the edge $\{k, l\}$ *belongs to* the underlying Hamiltonian cycle of the solution (permutation) s .

Once a distance function $d : E \mapsto \mathbb{R}_+$ is given, the (*total traveling*) *cost* $f(s)$ of a feasible solution $s = (i_1, i_2, \dots, i_n) \in S$ is then calculated by

$$f(s) := \sum_{j=1}^{n-1} d(i_j, i_{j+1}) + d(i_n, i_1). \quad (1)$$

We denote by $S^* \subseteq S$ the set of feasible solutions (Hamiltonian cycles) that minimize the cost (1).

3.2. Details of the CE variant

The CE variant we consider in the analysis completely inherits the structure of Algorithm 1, and additionally employs a component from *MMAS*.

We now formalize the sampling distribution, and define Steps 3 and 5 in more detail. As mentioned, we represent a sampling distribution (a product distribution on \mathcal{A}^n) for the TSP by a matrix $\mathbf{\Pi} = (\pi_{i,j})_{n \times n}$, such that

- a) $\sum_{j=1}^n \pi_{i,j} = 1$, for all $i = 1, \dots, n$,
- b) $\pi_{i,i} = 0$ for all $i = 1, \dots, n$,
- c) $\pi_{i,j} = \pi_{j,i}$ for each edge $\{i, j\} \in E$.

For each edge $\{i, j\} \in E$, $\pi_{i,j}$ reflects the probability that a Hamilton cycle continues with vertex j when it is in vertex i . In the sequel, we write the sampling distribution $\mathbf{\Pi}_t$ in iteration t as $(\pi_{i,j}^t)_{n \times n}$, where the superscript t of $\pi_{i,j}^t$ indicates the iteration. The *initial distribution* $\mathbf{\Pi}_0 = (\pi_{i,j}^0)_{n \times n}$ is, without loss of generality, set to be the uniform distribution, i.e., $\pi_{i,j}^0 = \pi_{j,i}^0 = \frac{1}{n-1}$ for all edges $\{i, j\} \in E$.

We shall consider two random solution generation methods, a *vertex-based random solution generation* and an *edge-based random solution generation*. Algorithm 2 lists the vertex-based random solution generation method. This method uses V as the ground set \mathcal{A} . A product distribution of \mathcal{A}^n is therefore represented as a matrix $\mathbf{\Pi} = (\pi_{i,j})_{n \times n}$ satisfying a)-c) above, i.e., each row of $\mathbf{\Pi}$ represents a sampling distribution on $\mathcal{A} = V$. Directly sampling from $\mathbf{\Pi}$ may produce infeasible solutions from $\mathcal{A}^n - S$. To avoid that, Algorithm 2 starts with a randomly fixed initial node, and then sequentially extends a partial solution with an unvisited vertex until a complete permutation is obtained. This method is efficient and rather popular in practice, see, e.g., [39] and [4]. Here, “ $s + (v)$ ” means that appends a vertex v to the end of a partial solution s .

The edge-based random solution generation is listed in Algorithm 3. The idea is from [40]. This method considers edge set E as the ground set \mathcal{A} . A feasible solution is then a sequence of edges that form a Hamiltonian cycle, i.e. $S \subseteq E^n$. To unify the notation of feasible solutions, Algorithm 3 translates its outcomes into permutations. As the actual ground set is E , a product distribution is an $n \times \frac{n(n-1)}{2}$ matrix such that each row is a marginal specifying a sampling distribution on E . Algorithm 3 only considers those with identical marginals, a product distribution can be therefore fully characterized by one of its marginals and is therefore again represented by an $n \times n$ matrix $\mathbf{\Pi} = (\pi_{i,j})_{n \times n}$ as above. An edge $\{i, j\} \in E$ is then sampled from $\mathbf{\Pi}$ with probability $(\pi_{i,j} + \pi_{j,i}) / \sum_{k=1}^n \sum_{l=1}^n \pi_{k,l} = 2\pi_{i,j}/n$ since each row of $\mathbf{\Pi}$ sums up to 1. A random sequence $\in E^n$ is generated by independently sampling from $\mathbf{\Pi}$ n times. To avoid infeasible solutions, Algorithm 3 considers in every sampling only edges that are *admissible* by the edges selected before. Given a set \mathcal{B} of edges such that the subgraph (V, \mathcal{B}) does neither contain a

Algorithm 2 Vertex-based random solution generation

Input:

a distribution $\mathbf{\Pi} = (\pi_{i,j})_{n \times n}$

Output:

a permutation of $1, 2, \dots, n$

- 1: $s = \emptyset$, and $V_{unvisited} = V$;
- 2: randomly select v from V , $s = s + (v)$, and $V_{unvisited} = V_{unvisited} - \{v\}$;
- 3: **while** ($|V_{unvisited}| \neq \emptyset$) **do**
- 4: select a random vertex v' from $V_{unvisited}$ with a probability

$$\mathbf{P}[v' \mid s] = \frac{\pi_{v,v'}}{\sum_{k \in V_{unvisited}} \pi_{v,k}}; \quad (2)$$

- 5: set $s = s + (v')$, $V_{unvisited} = V_{unvisited} - \{v'\}$;
 - 6: $v = v'$;
 - 7: **end while**
 - 8: **return** s ;
-

cycle nor a vertex of degree ≥ 3 , an edge $e' \in E$ is said to be admissible by \mathcal{B} if and only if the subgraph $(V, \mathcal{B} \cup \{e'\})$ still does neither contain a cycle nor a vertex of degree ≥ 3 . We denote by $B_{admissible}$ the set of edges $\notin \mathcal{B}$ that are admissible by \mathcal{B} .

The N random solutions $\mathbf{X}_t^{(1)}, \dots, \mathbf{X}_t^{(N)}$ in iteration t are then generated by N runs of Algorithm 2 or Algorithm 3 with *the current distribution* $\mathbf{\Pi}_t = (\pi_{i,j}^t)_{n \times n}$. The empirical distribution $\mathbf{W}_t = (w_{i,j}^t)_{n \times n}$ is then calculated from the M elite solutions by setting

$$w_{i,j}^t = \frac{\sum_{k=1}^M \mathbb{1}_{\{e' \in E \mid e' \in \mathbf{X}_t^{[k]}\}}(\{i, j\})}{M}, \quad (4)$$

where $\mathbb{1}_A(\cdot)$ is the indicator function of set $A = \mathbf{X}_t^{[k]}$ for each $\{i, j\} \in E$. The next distribution $\mathbf{\Pi}_{t+1} = (\pi_{i,j}^{t+1})_{n \times n}$ is therefore obtained as

$$\pi_{i,j}^{t+1} = (1 - \rho)\pi_{i,j}^t + \rho w_{i,j}^t \quad (5)$$

for each $\{i, j\} \in E$.

We continue with the suggestions made in [32]. In the CE variant, we shall use a moderately large N and a large $\rho = 1$. To fully use the best elite

Algorithm 3 Edge-based random solution generation

Input:

a distribution $\Pi = (\pi_{i,j})_{n \times n}$

Output:

a permutation of $1, 2, \dots, n$

- 1: $\mathcal{B} = \emptyset, B_{admissible} = E$;
- 2: **while** $(|\mathcal{B}| \leq n - 1)$ **do**
- 3: select an edge $\{i, j\}$ from $B_{admissible}$ with a probability

$$\mathbf{P}[e \mid s] = \frac{\pi_{i,j} + \pi_{j,i}}{\sum_{\{k,l\} \in B_{admissible}} \pi_{k,l} + \pi_{l,k}}; \quad (3)$$

- 4: set $\mathcal{B} = \mathcal{B} \cup \{\{i, j\}\}$;
 - 5: update $B_{admissible}$;
 - 6: **end while**
 - 7: let $s = (1, i_2, i_3, \dots, i_n)$ with $\{1, i_2\}, \{i_j, i_{j+1}\} \in \mathcal{B}$ for $j = 3, \dots, n-1$;
 - 8: **return** s ;
-

solutions, we take $M = 1$. To prevent premature convergence (i.e., a possible stagnation at a non-optimal solution), we use an add-on from \mathcal{MMAS} [10], called *max-min calibration*, in the construction of Π_{t+1} . We choose a lower bound $\pi_{\min} \in (0, 1)$ and an upper bound $\pi_{\max} \in (0, 1)$, and, after applying (5), adjust Π_{t+1} by

$$\pi_{i,j}^{t+1} = \begin{cases} \pi_{\min} & \text{if } \pi_{i,j}^{t+1} < \pi_{\min}, \\ \pi_{i,j}^{t+1} & \text{if } \pi_{i,j}^{t+1} \in [\pi_{\min}, \pi_{\max}], \\ \pi_{\max} & \text{if } \pi_{i,j}^{t+1} > \pi_{\max}, \end{cases} \quad (6)$$

for any edge $\{i, j\} \in E$. Note that the max-min calibration is the *only* step that does not occur in the general CE (i.e., Algorithm 1).

This setting turns CE into a \mathcal{MMAS} . The main difference is that in our algorithm, only the iteration-best solution (i.e., $\mathbf{X}_t^{[1]}$) is allowed to change the ‘pheromones’ Π_t , i.e., the so-called *iteration-best reinforcement*. Iteration-best reinforcement is seldom used in theoretical runtime analysis, although Stützle and Hoos [10] indicated in an empirical study that the practical performance of iteration-best reinforcement is comparable to best-so-far reinforcement.

4. Main results

We shall now analyze the stochastic runtime of our two different random solution generation methods for two classes of TSP instances that have been well studied in the literature.

4.1. Stochastic runtime analysis for simple instances

We first consider a class of simple TSP instances that is defined by the following distance function $d : E \rightarrow \mathbb{R}$ on a graph with n vertices.

$$d(\{i, j\}) = \begin{cases} 1 & \text{if } \{i, j\} = \{i, i+1\} \text{ for each } i = 1, 2, \dots, n-1, \\ 1 & \text{if } \{i, j\} = \{n, 1\}, \\ n & \text{otherwise.} \end{cases} \quad (7)$$

Obviously, TSP instances with this distance function have a unique optimal solution $s^* = (1, 2, \dots, n)$ (in the sense of the underlying Hamiltonian cycle), and s^* has a cost of n . The cost of an arbitrary feasible solution s equals $k + (n - k) \cdot n$, where k is the number of edges $\in s$ that are also in s^* . We shall refer to these instances as G_1 in the sequel.

The class G_1 has been used in [19] and [40] for analyzing the expected runtime of variants of *MMAS*. Zhou [19] proved that the $(1 + 1)$ MMAA algorithm has an expected runtime of $O(n^6 + \frac{n \ln n}{\rho})$ on G_1 in the case of non-visibility (i.e., without the greedy distance information in the sampling), and has an expected runtime of $O(n^5 + \frac{n \ln n}{\rho})$ in the case of visibility (i.e., with considering the greedy distance information in the sampling). Kötzing et al [40] continued the study in [19]. They investigated the expected runtime of $(1 + 1)$ MMAA and its variant $\text{MMAS}_{\text{Arb}}^*$ on G_1 and other TSP instances on which both $(1 + 1)$ MMAA and $\text{MMAS}_{\text{Arb}}^*$ have exponential expected runtime. $\text{MMAS}_{\text{Arb}}^*$ differs with $(1 + 1)$ MMAA only in the random solution generation. $\text{MMAS}_{\text{Arb}}^*$ uses Algorithm 3 as its random solution generation method, while $(1 + 1)$ MMAA used Algorithm 2. Kötzing et al [40] proved that $\text{MMAS}_{\text{Arb}}^*$ has an expected runtime of $O(n^3 \ln n + \frac{n \ln n}{\rho})$ on G_1 .

Theorem 1 shows a stochastic runtime of $O(n^{6+\epsilon})$ for the CE variant with the add-on, i.e., Algorithm 1 with max-min calibration (6), the vertex-based random solution generation, and a stochastic runtime of $O(n^{4+\epsilon})$ for the edge-based random solution generation. These results are comparable with the above known expected runtimes. Although we are not able to get strictly superior runtimes, our results are actually stronger and more informative.

Theorem 1 (Stochastic runtime of Algorithm 1 with max-min calibration on G_1). Assume that we set $M = 1$, $\rho = 1$, and use Algorithm 1 with the max-min calibration (6) for the values $\pi_{\min} = \frac{1}{n(n-2)}$, $\pi_{\max} = 1 - \frac{1}{n}$. Then

- a) if we use the vertex-based random solution generation method (Algorithm 2), and take a sample size $N = \Omega(n^{5+\epsilon})$ for any constant $\epsilon \in (0, 1)$, then with a probability at least $1 - e^{-\Omega(N/n^5)}$ the optimal solution s^* can be found within n iterations;
- b) if we use the edge-based random solution generation method (Algorithm 3), and take a sample size $N = \Omega(n^{3+\epsilon})$ for a constant $\epsilon \in (0, 1)$, then with a probability at least $1 - e^{-\Omega(N/n^3)}$, the optimal solution can be found within n iterations.

Proof. Recall that $s^* = (1, 2, \dots, n)$ is the unique optimal solution to G_1 , and the more edges from s^* a solution contains, the smaller cost it has. We define for each iteration $1 \leq t \leq n-1$ a random event \mathcal{E}_t by the following properties:

- i) if $|\mathbf{X}_{t-1}^{[1]} \cap s^*| = k \leq n-3$, then $|\mathbf{X}_t^{[1]} \cap s^*| \geq k+1$;
- ii) if $|\mathbf{X}_{t-1}^{[1]} \cap s^*| = n-2$, then $\mathbf{X}_t^{[1]} = s^*$;
- iii) if $\mathbf{X}_{t-1}^{[1]} = s^*$, then $\mathbf{X}_t^{[1]} = s^*$.

Here $\mathbf{X}_t^{[1]} \cap s^*$ is the set of common edges in s^* and $\mathbf{X}_t^{[1]}$. Obviously, $\bigcap_{t=1}^{n-1} \mathcal{E}_t$ implies that s^* is found within n iterations, since s^* has only n edges. We must inspect the probabilities of $\bigcap_{t=1}^{n-1} \mathcal{E}_t$ for the two different random solution generation methods.

Note that

$$\begin{aligned} \mathbf{P} \left[\bigcap_{t=1}^{n-1} \mathcal{E}_t \right] &= \sum_{s \in S} \mathbf{P}[\mathbf{X}_0^{[1]} = s] \cdot \mathbf{P} \left[\bigcap_{t=1}^{n-1} \mathcal{E}_t \mid \mathbf{X}_0^{[1]} = s \right] \\ &= \sum_{s \in S} \mathbf{P}[\mathbf{X}_0^{[1]} = s] \cdot \mathbf{P}[\mathcal{E}_1 | \mathbf{X}_0^{[1]} = s] \cdot \prod_{t=2}^{n-1} \mathbf{P}[\mathcal{E}_t \mid \bigcap_{k=1}^{t-1} \mathcal{E}_k, \mathbf{X}_0^{[1]} = s]. \end{aligned} \quad (8)$$

We will show below that for any $s \in S$ and each of the random solution generation method, the probability

$$\mathbf{P}[\mathcal{E}_1 | \mathbf{X}_0^{[1]} = s] \cdot \prod_{t=2}^{n-1} \mathbf{P}[\mathcal{E}_t \mid \bigcap_{k=1}^{t-1} \mathcal{E}_k, \mathbf{X}_0^{[1]} = s] \quad (9)$$

is bounded from below by an overwhelming probability that does not depend on s . This proves the Theorem.

Before we bound the probability (9), let us inspect the distribution $\mathbf{\Pi}_t$ more closely. Since $M = 1$, we only use the iteration-best solution $\mathbf{X}_t^{[1]}$ to build the empirical distribution $\mathbf{W}_t = (w_{i,j}^t)_{n \times n}$ for each iteration $t \in N$. As we set $\rho = 1$ and employ the max-min calibration (6), we have

$$\pi_{i,j}^{t+1} = \pi_{j,i}^{t+1} = \begin{cases} \min\{1, \pi_{\max}\} = \pi_{\max} & \text{if edge } \{i, j\} \in \mathbf{X}_t^{[1]}, \\ \max\{0, \pi_{\min}\} = \pi_{\min} & \text{otherwise,} \end{cases} \quad (10)$$

for every edge $\{i, j\} \in E$ and iteration $t \in \mathbb{N}$.

We only give a detailed proof for the lower bound of the first factor $\mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s]$ of (9). Each of the other factors has the same lower bound, which can be derived by a similar discussion for $\mathbf{X}_{t-1}^{[1]}$.

Observe that

$$\begin{aligned} \mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s] &= \mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s, |s \cap s^*| \leq n-3] \cdot \mathbf{P}[|\mathbf{X}_0^{[1]} \cap s^*| \leq n-3 \mid \mathbf{X}_0^{[1]} = s] \\ &\quad + \mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s, |s \cap s^*| = n-2] \cdot \mathbf{P}[|\mathbf{X}_0^{[1]} \cap s^*| = n-2 \mid \mathbf{X}_0^{[1]} = s] \\ &\quad + \mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s, s = s^*] \cdot \mathbf{P}[\mathbf{X}_0^{[1]} = s^* \mid \mathbf{X}_0^{[1]} = s], \end{aligned} \quad (11)$$

and

$$\begin{aligned} &\mathbf{P}[|\mathbf{X}_0^{[1]} \cap s^*| \leq n-3 \mid \mathbf{X}_0^{[1]} = s] + \mathbf{P}[|\mathbf{X}_0^{[1]} \cap s^*| = n-2 \mid \mathbf{X}_0^{[1]} = s] \\ &\quad + \mathbf{P}[\mathbf{X}_0^{[1]} = s^* \mid \mathbf{X}_0^{[1]} = s] = 1. \end{aligned}$$

To facilitate our discussion, we define the following five random events:

$$\mathcal{E}_{1,1} : \mathbf{X}_0^{[1]} = s, |s \cap s^*| \leq n-3;$$

$$\mathcal{E}_{1,2} : \mathbf{X}_0^{[1]} = s, |s \cap s^*| = n-2;$$

$$\mathcal{E}_{1,3} : \mathbf{X}_0^{[1]} = s, s = s^*;$$

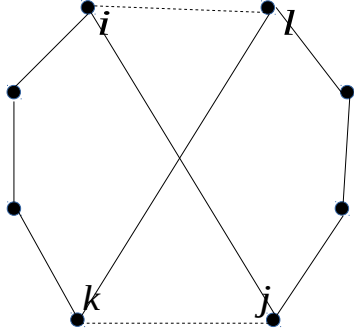
$$\mathcal{F}_{1,1} : |\mathbf{X}_1^{[1]} \cap s^*| \geq |\mathbf{X}_0^{[1]} \cap s^*| + 1;$$

$$\mathcal{F}_{1,2} : \mathbf{X}_1^{[1]} = s^*.$$

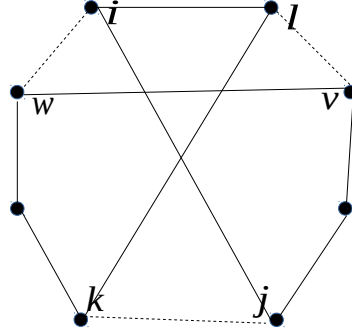
It is not difficult to see that, conditioned on $\mathcal{E}_{1,1}$, $\mathcal{E}_1 = \mathcal{F}_{1,1}$, and, conditioned on $\mathcal{E}_{1,2}$ or $\mathcal{E}_{1,3}$, $\mathcal{E}_1 = \mathcal{F}_{1,2}$. So we obtain with (11) that

$$\mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s] \geq \min\{\mathbf{P}[\mathcal{F}_{1,1} \mid \mathcal{E}_{1,1}], \mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,2}], \mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,3}]\}. \quad (12)$$

We will now derive a lower bound for the right-hand-side of (12) by inspecting two special operations that may reduce the total traveling cost of an iteration-best solution. A *2-exchange move* on a Hamiltonian cycle is an operation that removes two edges of the cycle and adds two (unique) new edges to obtain again a cycle. Figure 1a shows an example of a 2-exchange, in which edges $\{i, j\}, \{k, l\}$ are removed, and edges $\{i, l\}, \{k, j\}$ are added. A 2-exchange move that reduces the cost is called a *2-opt move*. A *3-exchange move* is an operation that removes three edges and adds three new edges to obtain again a cycle, see, e.g., Figure 1b. A 3-exchange that reduces the cost is called a *3-opt move*. Due to the $\{1, n\}$ -valued distance function of G_1 , a 2-opt move or a 3-opt move on $\mathbf{X}_0^{[1]} = s$ that happens in the sampling of iteration 1 implies that $\mathbf{X}_1^{[1]}$ contains more edges from s^* than $\mathbf{X}_0^{[1]}$.



(a) A 2-exchange operation



(b) A 3-exchange operation

Figure 1: Example for edge exchange operations

Zhou [19] has studied the probabilities that a 2-opt move or a 3-opt move occurs in the vertex-based random solution generation. With $\pi_{\min} = \frac{1}{n^2}$ and $\pi_{\max} = 1 - \frac{1}{n}$, Zhou [19] proved for (1+1) MMAA that with a probability of $\Omega(1/n^5)$, Algorithm 2 produces a random solution having more edges from s^* than x_t^* (the best solution found so far) provided that x_t^* is not optimal. Zhou [19] actually showed that if $x_t^* \neq s^*$, then there exists either a 2-opt move or a 3-opt move for x_t^* , and Algorithm 2 produces an arbitrary 2-exchange of x_t^* with a probability of $\Omega(1/n^3)$, and an arbitrary 3-exchange of x_t^* with a probability of $\Omega(1/n^5)$.

Here, we use $\pi_{\min} = \frac{1}{n(n-2)}$. We now assert that the above properties for

x_t^* also hold for $\mathbf{X}_t^{[1]}$. In order to produce a 2-exchange move of $\mathbf{X}_t^{[1]}$, one possible way is to add one of the new edges at the end of the vertex-based random solution generation. Recall that in Algorithm 2, the probability (2) to select a continuing edge $\{i, j\}$ is always bounded from below by $\pi_{i,j}^t$ (or, equivalently, $\pi_{j,i}^t$) for each iteration $t \in \mathbb{N}$, since each row of $\mathbf{\Pi}_t$ sums up to 1. So an arbitrary 2-exchange move of $\mathbf{X}_t^{[1]}$ happens in any of the N independent draws of iteration $t + 1$ with a probability larger than

$$\frac{1}{n} \cdot \frac{1}{n(n-2)} \cdot \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{1}{e \cdot n^3},$$

where $e \approx 2.71828$ is Euler's number, $\frac{1}{n}$ represents the probability to select the starting vertex, $\frac{1}{n(n-2)}$ is the lower bound of the probability to select another new edge, and $1 - \frac{1}{n}$ is the common lower bound of the probability to select one of the the remaining $n - 2$ edges from $\mathbf{X}_t^{[1]}$.

Similarly, a 3-exchange move can be produced by adding one of the three new edges at the end in the vertex-based random solution generation, and this happens with a probability at least $\frac{1}{e \cdot n^5}$. Actually, we can even show that, with the vertex-based random solution generation, a k -exchange move (removing k edges and adding k new edges) of $\mathbf{X}_t^{[1]}$ happens with a probability at least

$$\frac{1}{e \cdot n^{2k-1}}$$

for every integer $k = 2, 3, \dots, n$.

Conditioned on $\mathcal{E}_{1,1}$ (or $\mathcal{E}_{1,2}$), $s \neq s^*$, and therefore a 2-opt move or a 3-opt move of $\mathbf{X}_0^{[1]}$ exists, see [19] (or the proof of Theorem 2 b)). By the above arguments, in the N many independent applications of Algorithm 2, the probability that at least one of $\mathbf{X}_1^{(1)}, \dots, \mathbf{X}_1^{(N)}$ is a 2-opt or 3-opt move of $\mathbf{X}_0^{[1]}$ is at least

$$1 - \left(1 - \frac{1}{e \cdot n^5}\right)^N \geq 1 - e^{-\frac{N}{e \cdot n^5}}.$$

And, if this is the case, $f(\mathbf{X}_1^{[1]})$ will be strictly smaller than $f(\mathbf{X}_0^{[1]})$. This, in turn, implies that

$$\mathbf{P}[\mathcal{F}_{1,1} \mid \mathcal{E}_{1,1}] \geq 1 - e^{-\frac{N}{e \cdot n^5}} \text{ and } \mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,2}] \geq 1 - e^{-\frac{N}{e \cdot n^5}}, \quad (13)$$

where we observe that there exists a 2-opt move that turns s into s^* if $|s \cap s^*| = n - 2$.

Conditioned on $\mathcal{E}_{1,3}$, $\mathbf{X}_0^{[1]} = s = s^*$. In this case, the probability of the event that a random solution sharing the same Hamiltonian cycle with s^* occurs among $\mathbf{X}_1^{(1)}, \dots, \mathbf{X}_1^{(N)}$, is bounded from below by

$$(1 - \frac{1}{n})^{n-1} \geq \frac{1}{e}.$$

Therefore,

$$\mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,3}] \geq 1 - (1 - 1/e)^N = 1 - e^{-N \ln \frac{e}{e-1}}. \quad (14)$$

Actually, (14) is also a lower bound for the probability of event that $\mathbf{X}_{t+1}^{[1]}$ contains only better or equally good solutions as $\mathbf{X}_t^{[1]}$, for any arbitrarily fixed $t \in \mathbb{N}$.

Combining (12), (13) and (14), we obtain

$$\mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s] \geq 1 - e^{-\frac{N}{e \cdot n^5}},$$

which completes the proof of *a*).

The behavior of the edge-based random solution generation is comprehensively studied in [40]. Kötzing et al [40] proved for MMAS_{Arb}^* and a constant $k \in O(1)$ that, with a probability of $\Omega(1)$, Algorithm 3 produces a random solution that is obtained by a k -exchange move from the best solution found so far. We now show that this still holds here.

Recall that in each iteration t , either $\pi_{i,j}^t = \pi_{j,i}^t = \pi_{\min}$ or $\pi_{i,j}^t = \pi_{j,i}^t = \pi_{\max}$ for any edge $\{i, j\} \in E$. For convenience, we will call an edge $\{i, j\} \in E$ with $\pi_{i,j}^t = \pi_{j,i}^t = \pi_{\max}$ a *high* edge, and otherwise a *low* edge. We first inspect the probability of event that Algorithm 3 chooses a high edge in an arbitrary fixed step conditioned on the event that $l \leq \sqrt{n}$ many low edges have been chosen in some l steps before this step. A similar discussion of this probability can be found in [40].

Let $m \geq 1$ be the number of edges that still need to be added to obtain a solution. We now estimate the numbers of admissible high and low edges in this step. We can add at most m edges, and every of the l low edges blocks at most 3 of the $m+l$ remaining high edges (two because there may already be other edges at both ends, and one that may introduce a cycle). So at least $m+l-3l = m-2l \geq m-3l$ high edges are available for adding in this step. Of course, it may happen that there is no admissible high edges in this step. However, we are not interested in such a case. We consider only the case that

there exists at least one admissible high edge in this step, i.e. the number of admissible high edges in this step is at least $\max\{1, m - 3l\}$. Note also that the $n-m$ edges added before partition the subgraph of $G = (V, E)$ with vertices V and edges from the partial solution constructed so far into exactly m connected components (here, we see an isolated vertex also as a connected component). For any two of the components, there are at most 4 admissible edges connecting them. Therefore, there are at most $\min\{4\binom{m}{2}, \binom{n}{2}\}$ many admissible low edges. Observing $l \leq \sqrt{n}$, the probability of choosing a high edge in this step is bounded from *below* by

$$1 - \frac{\min\{4\binom{m}{2}, \binom{n}{2}\}}{\max\{1, m - 3l\}} \frac{\pi_{\min}}{\pi_{\max}} \geq \begin{cases} 1 - \frac{2m^2}{(m-3l)n(n-2)} = 1 - O(\frac{1}{n}) & \text{if } m > 3\sqrt{n}, \\ 1 - \frac{12}{n-2} = 1 - O(\frac{1}{n}) & \text{if } m \leq 3\sqrt{n}, \end{cases} \quad (15)$$

where the last inequality is obtained by observing that

$$\min\{4\binom{m}{2}, \binom{n}{2}\} \leq \min\{2m^2, \binom{n}{2}\} \leq 2m^2,$$

and $\tau_{\max} = 1 - 1/n$, $\pi_{\min} = \frac{1}{n(n-2)}$.

With the above fact, we are now able to show that, for any $t \in \mathbb{N}$ and any fixed $k \in O(1)$, the probability of the event that Algorithm 3 produces a k -exchange of $\mathbf{X}_t^{[1]}$ in any of the N many independent draws in iteration $t+1$, is $\Omega(1)$. Here, we shall use a different proof from the one presented by Kötzing et al [40], which appears to us as problematic.

Let $k \in O(1)$ be arbitrarily fixed, and \mathcal{M} be the set of all k -element subsets of $\{1, 2, \dots, n/2\}$ (where we assume without loss of generality that n is even). Obviously, $|\mathcal{M}| \in \Theta(n^k)$ since $k \in O(1)$. Let $\mathbb{M} \in \mathcal{M}$ be an arbitrarily fixed k -element subset. The probability of the event that Algorithm 3 selects k new edges (low edges) in steps $i \in \mathbb{M}$ and $n - k$ edges (high edges) from $\mathbf{X}_t^{[1]}$ in other steps, is bounded from below by

$$\begin{aligned} & (1 - O(\frac{1}{n}))^{n-k} \prod_{i \in \mathbb{M}} \frac{((\binom{n-i+1}{2}) - (n-i+k+1))\pi_{\min}}{n(n-1)\pi_{\min} + (n-i+k+1)\pi_{\max}} \\ & \geq (1 - O(\frac{1}{n}))^{n-k} \pi_{\min}^k \prod_{i \in \mathbb{M}} \frac{((\binom{n-i+1}{2}) - (n-i+k+1))}{1 + (n-i+k+1)} \\ & \geq (1 - O(\frac{1}{n}))^{n-k} \pi_{\min}^k \prod_{i \in \mathbb{M}} \frac{(n^2/2 - (n+k+2))}{n+k+2} \\ & = \Theta(\frac{1}{n^k}), \end{aligned} \quad (16)$$

where $1 - O(1/n)$ is a lower bound for the probability of selecting an edge from $\mathbf{X}_t^{[1]}$. In each step $i \in \mathbb{M}$, the edges chosen before partition the graph into $n - i + 1$ many connected components, and for any two of the components there exists at least 2 edges connecting them without introducing a cycle. Hence, there are at least $\binom{n-i+1}{2}$ many admissible edges in each step $i \in \mathbb{M}$. Notice also that the number of admissible high edges in this case is at most $n - i + k + 1$ ($n - i + k + 1$ is the maximal number of high edges that have not been chosen before). Therefore, each factor

$$\frac{(\binom{n-i+1}{2} - (n - i + k + 1))\pi_{\min}}{n(n-1)\pi_{\min} + (n - i + k + 1)\pi_{\max}}$$

of (16) is just the lower bound of the probability for choosing an admissible edge not belonging to $\mathbf{X}_t^{[1]}$ in a step $i \in \mathbb{M}$.

As a result, the probability of the random event that Algorithm 3 produces a k -exchange of $\mathbf{X}_t^{[1]}$ with $k \in O(1)$ in any of the N independent draws in iteration $t+1$ is bounded from below by

$$|\mathcal{M}| \cdot \Theta\left(\frac{1}{n^k}\right) = \Theta(n^k) \cdot \Theta\left(\frac{1}{n^k}\right) \in \Omega(1),$$

since new edges can also be added in steps $l \geq n/2$.

We will now finish the proof of b). Observe that any two k -exchanges are generated with the same probability by Algorithm 3. So we can conclude that, for any $k \in O(1)$, if there exists a k -opt move of $\mathbf{X}_t^{[1]}$, then with a probability of $\Omega(1/n^k)$, Algorithm 3 produces the k -opt move in any of the N many independent draws in iteration $t+1$.

Observe also that when $\mathbf{X}_t^{[1]} \neq s^*$, then there exist a 2-opt or 3-opt move of $\mathbf{X}_t^{[1]}$ for any $t \in \mathbb{N}$, see [19] (or the proof of Theorem 2 b)). Therefore, we have

$$\mathbf{P}[\mathcal{F}_{1,1} \mid \mathcal{E}_{1,1}] \geq 1 - e^{-\frac{N}{e \cdot n^3}} \text{ and } \mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,2}] \geq 1 - e^{-\frac{N}{e \cdot n^3}}.$$

Now, we consider the probability of $\mathcal{F}_{1,2}$ conditioned on $\mathcal{E}_{1,3}$. With (15), we can easily show that the probability of producing a random solution that shares the same Hamiltonian cycle with $\mathbf{X}_0^{[1]} = s$ in one application of Algorithm 3 in iteration 1 is $\Omega(1)$. Therefore, we obtain that

$$\mathbf{P}[\mathcal{F}_{1,2} \mid \mathcal{E}_{1,3}] \geq 1 - (1 - \Omega(1))^N = 1 - e^{-\Omega(N)}.$$

This, in turn, implies that

$$\mathbf{P}[\mathcal{E}_1 \mid \mathbf{X}_0^{[1]} = s] \geq 1 - e^{-\frac{N}{e \cdot n^3}},$$

which completes the proof of b). \square

The stochastic runtimes of Theorem 1 are derived for a relatively large sample size, namely $N = \Omega(n^{5+\epsilon})$ and $N = \Omega(n^{3+\epsilon})$. Actually, Theorem 1 may still hold for a smaller sample size. Theorem 2 partially asserts this. It states that the total number of iterations required to reach the optimal solution for both generation schemes may increase considerably if a smaller sample size is used. However, the stochastic runtime does not increase. Interestingly, one can obtain a smaller stochastic runtime with a small sample size for the edge-based random solution generation.

Theorem 2 (Stochastic runtime of Algorithm 1 on G_1 for a small sample size). *Assume the conditions in Theorem 1, but set $N = \Omega(n^\epsilon)$ for any $\epsilon \in (0, 1)$. Then:*

- a) *For the vertex-based random solution generation, Algorithm 1 finds the optimal solution s^* within n^6 many iterations with a probability of $1 - e^{-\Omega(N)}$.*
- b) *For the edge-based random solution generation, Algorithm 1 finds the optimal solution s^* within $n^3 \ln n$ many iterations with a probability of $1 - e^{-\Omega(N)}$.*

Proof of Theorem 2. a) We consider two random events in the first n^6 iterations. The first random event \mathcal{E}_1 says that $f(\mathbf{X}_{t+1}^{[1]}) \leq f(\mathbf{X}_t^{[1]})$ for any $0 \leq t \leq n^6$. The second random event \mathcal{E}_2 says that for each $k = 0, 1, 2, \dots, n-1$, there exists an iteration $t \in [k \cdot n^5, (k+1) \cdot n^5)$ such that $|\mathbf{X}_{t+1}^{[1]} \cap s^*| > |\mathbf{X}_t^{[1]} \cap s^*|$, or the optimal solution is met within time window $[k \cdot n^5, (k+1) \cdot n^5)$.

Obviously, $\mathcal{E}_1 \cap \mathcal{E}_2$ implies that the optimal solution is found within n^6 iterations. Note that

$$\mathbf{P}[\mathcal{E}_1 \cap \mathcal{E}_2] \geq \mathbf{P}[\mathcal{E}_1] + \mathbf{P}[\mathcal{E}_2] - 1.$$

Recall that, in each iteration $t+1$ with $0 \leq t \leq n^6 - 1$, the probability of producing a random solution sharing the same Hamiltonian cycle with $\mathbf{X}_t^{[1]}$ in one application of Algorithm 2 is $\Omega(1)$. Therefore,

$$\mathbf{P}[\mathcal{E}_1] \geq (1 - (1 - \Omega(1))^N)^{n^6} = 1 - e^{-\Omega(N)}$$

when $N = \Omega(n^\epsilon)$.

Recall also that if $\mathbf{X}_t^{[1]} \neq s^*$, then $|\mathbf{X}_{t+1}^{[1]} \cap s^*| > |\mathbf{X}_t^{[1]} \cap s^*|$ has a probability of at least

$$1 - (1 - \Omega(1/n^5))^N.$$

Therefore, within n^5 consecutive iterations, the probability of the event that the optimal solution does not occur and no strict improvement happens, is bounded from above by

$$(1 - \Omega(1/n^5))^{N \cdot n^5}.$$

This implies that

$$\mathbf{P}[\mathcal{E}_2] \geq (1 - (1 - \Omega(1/n^5))^{N \cdot n^5})^n = 1 - e^{-\Omega(N)}.$$

Therefore,

$$\mathbf{P}[\mathcal{E}_1 \cap \mathcal{E}_2] \geq 1 - e^{-\Omega(N)}.$$

b) can be proved in a similar way as *a)*. Recall from that proof that, in each iteration $t \geq 1$, Algorithm 3 reproduces $\mathbf{X}_{t-1}^{[1]}$ with a probability of $\Omega(1)$. Therefore, the probability of the event \mathcal{E}_1 (defined in *a)*) that $|\mathbf{X}_t^{[1]} \cap s^*| \geq |\mathbf{X}_{t-1}^{[1]} \cap s^*|$ holds in each of the first $n^3 \ln n$ many iterations, is overwhelmingly large, i.e., $1 - e^{-\Omega(N)}$.

Here, we redefine the random event \mathcal{E}_2 as follows: for each $t \leq n^3 \ln n$, if $|\mathbf{X}_t^{[1]} \cap s^*| = n - k$ for some $k > 0$, then there exists some t_1 with $t + 1 \leq t_1 \leq t + \lfloor n^3/k \rfloor$ and $|\mathbf{X}_{t_1}^{[1]} \cap s^*| \geq n - k + 1$. Obviously, an overwhelming probability of $\mathcal{E}_1 \cap \mathcal{E}_2$ again implies the conclusion. Now, we only need to analyze the probability of \mathcal{E}_2 .

Kötzing et al [40] showed for MMAS_{Arb}^* that if the best solution s_t^* found so far has $n - k$ many edges from s^* , then the probability of the event that s_{t+1}^* has at least $n - k + 1$ many edges from s^* , is in $\Omega(k/n^3)$. We shall use a different but simpler proof to show that this also holds in our case of iteration-best reinforcement.

We first show that when $|\mathbf{X}_t^{[1]} \cap s^*| = n - k$ with $k > 0$, then there exists a 2-opt move or a 3-opt move for $\mathbf{X}_t^{[1]}$. Assume that $\mathbf{X}_t^{[1]}$ contains exactly $n - k$ edges from s^* for some integer $k > 0$. Let $e^* = \{i, i + 1\}$ be an edge in s^* but *not* in $\mathbf{X}_t^{[1]}$. Note that each node of the graph is exactly incident to two edges of s^* and $\mathbf{X}_t^{[1]}$, respectively. Therefore there exists an edge $e_0 \in \mathbf{X}_t^{[1]}$ incident to i , an edge $e'_0 \in \mathbf{X}_t^{[1]}$ incident to $i + 1$, and e_0, e'_0 are not in s^* . Figure 2 shows an example, where e_0 is either $\{i, u\}$ or $\{i, v\}$, and e'_0 is either $\{i + 1, w\}$ or $\{i + 1, y\}$. If $e_0 = \{i, u\}$ and $e'_0 = \{i + 1, w\}$ or if $e_0 = \{i, v\}$ and $e'_0 = \{i + 1, y\}$,

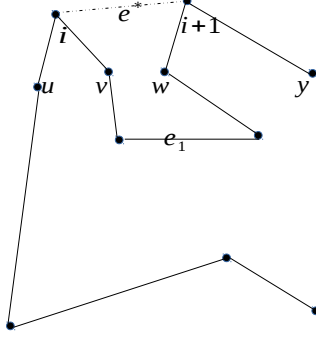


Figure 2: Demonstration of adding a new edge. The solid edges represent the cycle $\mathbf{X}_t^{[1]}$.

then there exists a 2-opt move of $\mathbf{X}_t^{[1]}$ which removes e_0, e'_0 of distance n and adds e^* and another edge (either $\{u, w\}$ or $\{v, y\}$) of distance at most $n + 1$ together. If $e_0 = \{i, u\}, e'_0 = \{i+1, y\}$ or $e_0 = \{i, v\}, e'_0 = \{i+1, w\}$, there is a 3-opt move of $\mathbf{X}_t^{[1]}$ which removes e_0, e'_0 , and an edge $e_1 \notin s^*$, and adds edge e^* and another two edges, this replacing 3 edges of distance n by 3 edges of distance at most $2n + 1$ together. Here, observe the fact that adding e^* to $\mathbf{X}_t^{[1]}$ and removing e_0, e'_0 from $\mathbf{X}_t^{[1]}$ results in graph containing a cycle, and there must be an edge $e_1 \in \mathbf{X}_t^{[1]}$ on that cycle that does not belong to s^* . We choose this edge as the edge e_1 . Therefore, for each e^* of the k remaining edges in s^* that are not in $\mathbf{X}_t^{[1]}$, there exists a 2-opt or 3-opt move of $\mathbf{X}_t^{[1]}$ that adds e^* .

In the proof of Theorem 1 b), we have shown that, for any $l \in O(1)$, the probability of producing a l -exchange of the iteration-best solution $\mathbf{X}_t^{[1]}$ by Algorithm 3 in iteration $t + 1$ is $\Omega(1)$. Since any two l -exchanges are produced with the same probability, the probability of producing a *particular* l -exchange in iteration $t + 1$ is $\Omega(1/n^l)$. As a result, Algorithm 3 produces for each edge $e^* \in s^* - \mathbf{X}_t^{[1]}$ a 2-opt or 3-opt move of $\mathbf{X}_t^{[1]}$ that adds edge e^* with probability at least $\Omega(1/n^3)$.

Note that the generation of a 2-exchange (or a 3-exchange) with two newly added edges e_2, e_3 by Algorithm 3 includes two mutually exclusive cases ($3!$ cases for a 3-exchange): e_2 is chosen before e_3 , or e_3 is chosen before e_2 . It is not difficult to see that these two cases ($3!$ many cases for 3-exchange) have the same probability. Therefore, the probability of the *event*

that Algorithm 3 generates a 2-opt or 3-opt move of $\mathbf{X}_t^{[1]}$ that e^* as one of the newly added edges and selects e^* before the other newly added edges, is bounded from below by $\Omega(1/3!n^3) = \Omega(1/n^3)$. Since $\mathbf{X}_t^{[1]}$ has k such e^* and the corresponding k events are also mutually exclusive, we obtain that the probability that $\mathbf{X}_{t+1}^{[1]}$ has more edges from s^* than $\mathbf{X}_t^{[1]}$ if $\mathbf{X}_t^{[1]}$ has exactly $n - k$ many edges from s^* for a constant $k > 0$ is $\Omega(k/n^3)$

With the above fact, we can easily see that for any fixed $k = 2, \dots, n$ and any fixed $t \in \mathbb{N}$ such that $|\mathbf{X}_t^{[1]} \cap s^*| = n - k$, the probability of the event, that

$$|\mathbf{X}_t^{[1]} \cap s^*| \geq |X_{t+1}^{[1]} \cap s^*|, |X_{t+1}^{[1]} \cap s^*| \geq |X_{t+2}^{[1]} \cap s^*|, \dots, |X_{t+\lfloor n^3/k \rfloor - 1}^{[1]} \cap s^*| \geq |X_{t+\lfloor n^3/k \rfloor}^{[1]} \cap s^*|$$

is bounded from *above* by

$$(1 - \Omega(k/n^3))^{N \cdot n^3/k}.$$

Therefore, for any fixed $k = 2, \dots, n$ and any fixed $t \in \mathbb{N}$ with $|\mathbf{X}_t^{[1]} \cap s^*| = n - k$, the probability of the event, that there exists a $j = 0, \dots, \lfloor n^3/k \rfloor - 1$ such that

$$|X_{t+j+1}^{[1]} \cap s^*| > |X_{t+j}^{[1]} \cap s^*|,$$

is bounded from *below* by

$$1 - (1 - \Omega(k/n^3))^{N \cdot n^3/k} = 1 - e^{\Omega(N)}$$

since $N = n^\epsilon$. This completes the proof of *b*). \square

Comparing the stochastic runtimes in Theorem 2 with those in the corresponding literature, we see that iteration-best reinforcement is also a good strategy for efficiently finding an optimal solution, although this strategy is seldom employed in the theoretical runtime analysis of evolutionary algorithms. In the following, we shall further assert this on more involved TSP instances.

4.2. Stochastic runtime analysis for grid instances

Now, we consider more general TSP instances. Herein, the n vertices are positioned on an $m \times m$ grid for some integer $m \in \mathbb{N}_+$. The vertices are positioned in a way that no three of them are *collinear*. Figure 3 gives an example of such an instance where $m = 5$ and $n = 8$. The weight of an edge $\{l, k\} \in E$ in this case is defined as the usual Euclidean distance $d(l, k)$

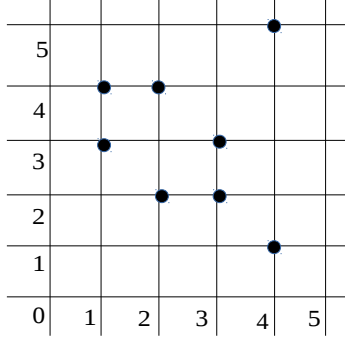


Figure 3: A grid instance

between vertex l and vertex k for every $l, k = 1, \dots, n$. In the sequel, we shall refer to these TSP instances as *grid instances*.

Grid instances have been studied in [41] and [28]. Sutton and Neumann [41] investigated the expected runtime of $(1+1)$ EA and RLS for these instances. As a continuation of [41], Sutton et al [28] further proved that the more extensive algorithm $(\mu + \lambda)$ EA finds an optimal solution for the instances *expectedly* in

$$O((\mu/\lambda)n^3m^5 + nm^5 + (\mu/\lambda)n^{4k}(2k-1)!)$$

many *iterations* if every of the λ selected parents is mutated by taking a random number of consecutive 2-exchange moves, and expectedly in

$$O((\mu/\lambda)n^3m^5 + nm^5 + (\mu/\lambda)n^{2k}(k-1)!)$$

iterations with a mixed mutation operator, where k denotes the number of vertices that are not on the boundary of the convex hull of V . Sutton et al [28] also studied general Euclidean TSP instances (without collinearity) and showed similar results in terms of the maximum distance value d_{\max} , the minimum distance value d_{\min} , k and the minimum angle in the triangles formed by the vertices.

Before we present our stochastic runtime, we summarize some structural properties of grid instances (some just follow from properties of general Euclidean instances). We say that two different edges $\{i, j\}$ and $\{k, l\}$ *intersect* with each other if there exists a point p such that $p \notin \{i, j, k, l\}$ locates on both of the two edges, see, e.g., Figure 4a. We say that a solution is

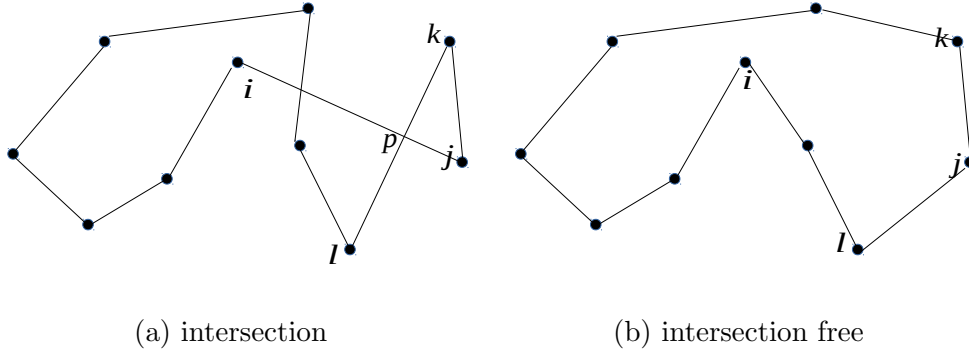


Figure 4: Example for intersections

intersection-free if the corresponding Hamiltonian cycle does not contain intersections, see, e.g., Figure 4b.

Obviously, the *triangle inequality* [47] holds for grid instances. Therefore, removing an intersection by a (unique) 2-exchange move in a solution strictly reduces the total traveling cost, see Figure 4a. Lemma 1 states the well known fact that an optimal solution of grid instances is intersection-free.

Lemma 1. *Optimal solutions of grid instances are intersection-free.*

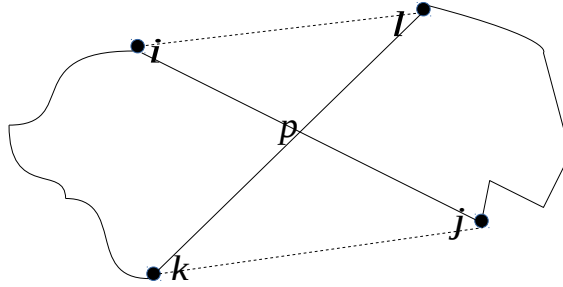


Figure 5: Example for a 2-opt move

We now *restrict* 2-opt moves to 2-exchange moves that remove an intersection. For example, removing edges $\{i, j\}, \{k, l\}$ in Figure 5 and adding new edges $\{i, l\}, \{k, j\}$ form such a 2-opt move. Lemma 2 below says that for

grid instances, removing one intersection may reduce the total traveling cost at least $\Omega(m^{-4})$ if it is applicable. We omit the simple proof here. Interested readers may refer to [28] for a proof.

Lemma 2. *If a feasible solution to a grid instance contains intersections, then removing the intersection can reduce the total traveling cost at least $\Omega(m^{-4})$.*

The *convex hull* $\mathfrak{V}(V)$ of the vertex set V is the smallest convex set in \mathbb{R}^2 that contains V . Its boundary is a convex polygon spanned by some vertices with possibly other vertices in the interior of that polygon. Let V^b denote the set of vertices on the boundary of $\mathfrak{V}(V)$. Figure 6 illustrates this.

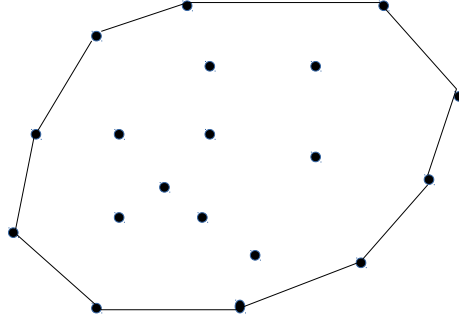


Figure 6: Example of a convex hull

Quintas and Supnick [48] proved that if a solution s is intersection-free, then the solution respects the hull-order, i.e., any two vertices in the subsequence of s induced by the boundary (the outer polygon) of $\mathfrak{V}(V)$ are consecutive in s if and only if they are consecutive on the boundary of $\mathfrak{V}(V)$. Therefore, if $V^b = V$, then every intersection-free solution is optimal.

Theorem 3 below analyzes the stochastic runtime of Algorithm 1 for grid instances for the case that $V = V^b$. It states that the stochastic runtime is $O(n^4 \cdot m^{5+\epsilon})$ for the vertex-based random solution generation, and $O(n^3 \cdot m^{5+\epsilon})$ for the edge-based random solution generation. These stochastic runtimes are very close to the expected runtime $O(n^3 \cdot m^5)$ for RLS reported by Sutton et al [41] and [28].

Theorem 3. *Consider a TSP instance with n vertices located on an $m \times m$ grid such that no three of them are collinear. Assume that $V^b = V$, that*

we apply the max-min calibration (6) with $\pi_{\max} = 1 - \frac{1}{n}$, $\pi_{\min} = \frac{1}{n(n-2)}$, $\rho = 1$, $M = 1$ and $N = \Omega(m^\epsilon)$ for some constant $\epsilon > 0$. Then:

- a) With an overwhelming probability of $1 - e^{-\Omega(N)}$, Algorithm 1 finds the optimal solution within at most $n^4 \cdot m^5$ iterations with the vertex-based random solution generation.
- b) With an overwhelming probability of $1 - e^{-\Omega(N)}$, Algorithm 1 finds an optimal solution within at most $n^3 \cdot m^5$ iterations with edge-based random solution generation.

Proof of Theorem 3. Note that under the conditions of Theorem 3, every intersection free solution is optimal. By Lemma 2, we know that a 2-opt move reduces the total traveling cost by $\Omega(m^{-4})$. Therefore, $n \cdot m^5$ many consecutive 2-opt moves turn a feasible solution into an optimal one, since the worst solution in this case has a total traveling cost smaller than $n \cdot m$ and the optimal solution has total traveling cost larger than n . Notice also that $m \geq n/2$, since the n vertices are positioned on the $m \times m$ grid and no three of them are collinear.

a) Since $m \geq n/2$ and $N = \Omega(m^\epsilon)$, one can show that the probability of the event that $f(\mathbf{X}_{t-1}^{[1]}) \geq f(X_t^{[1]})$ for each $t \leq n^4 \cdot m^5$ is at least

$$(1 - (1 - \Omega(1))^N)^{n^4 \cdot m^5} = 1 - e^{-\Omega(N)},$$

where $1 - (1 - \Omega(1))^N$ is a lower bound for the probability of the event that $\mathbf{X}_{t-1}^{[1]}$ is reproduced in iteration t , see proof of Theorem 1 a).

Recall that Algorithm 2 produces a 2-opt move for $\mathbf{X}_t^{[1]}$ with a probability of $\Omega(1/n^3)$. Therefore, for any iteration $t \in \mathbb{N}$ with $\mathbf{X}_t^{[1]}$ being non-optimal, the probability of the event that an optimal solution does not occur in the time window $[t, t+n^3-1]$ and no 2-opt move happens in the time window $[t+1, t+n^3]$ is bounded from above by

$$(1 - \Omega(\frac{1}{n^3}))^{n^3 \cdot N}.$$

Hence, for any iteration t with $\mathbf{X}_t^{[1]}$ being non-optimal, the probability of the event that an optimal solution occurs in the time window $[t, t+n^3-1]$ or least one 2-opt move happens in the time window $[t+1, t+n^3]$ is bounded from below by

$$1 - (1 - \Omega(\frac{1}{n^3}))^{n^3 \cdot N} = 1 - e^{-\Omega(N)}.$$

As a result, the probability of the event that an optimal solution is found within $n \cdot m^5$ iterations is bounded from below by

$$1 - e^{-\Omega(N)} + (1 - e^{-\Omega(N)})^{n \cdot m^5} - 1 = 1 - e^{-\Omega(N)}.$$

b) is similar to *a)*, except that the probability of producing a 2-opt move by Algorithm 3 is $\Omega(1/n^2)$ (see [40], or the proof of Theorem 1 *b)*). \square

Now, we consider the more interesting case that $|V - V^b| = k \in O(1)$. Note that we can turn an arbitrary intersection-free solution to an optimal solution only by rearranging the positions of those k interior points in that solution, and this requires at most k many consecutive jump moves (see [28] for a proof). A *jump move* $\delta_{i,j}$ transforms a solution into another solution by shifting positions i, j as follows. Solution s is transformed into solution $\delta_{i,j}(s)$ by moving the vertex at position i into position j while vertices at positions between i and j are shifted appropriately, e.g.,

$$\delta_{2,5}(i_1, i_2, i_3, i_4, i_5, i_6, i_7) = (i_1, i_3, i_4, i_5, i_2, i_6, i_7) \quad \text{and}$$

$$\delta_{5,2}(i_1, i_2, i_3, i_4, i_5, i_6, i_7) = (i_1, i_5, i_2, i_3, i_4, i_6, i_7).$$

It is not difficult to see that a jump move $\delta_{i,j}$ can be simulated by either a 2-exchange move (in the case that $|i-j|=1$) or a 3-exchange move (in all other cases). Therefore, we can actually turn an intersection-free solution into an optimal one by a sequence of at most k many consecutive 2-exchange or 3-exchange moves. Furthermore, a sequence of k many consecutive 2-exchange or 3-exchange moves can be simulated by a κ -exchange move with an integer $\kappa \leq 3k$. This means that any intersection-free solution can be turned into an optimal solution by a κ -exchange move with $\kappa \leq 3k$. We shall call such a κ -exchange move in the sequel a *3k-opt move*, although κ may be smaller than $3k$. Recall that a $3k$ -opt move is produced with a probability of $\Omega(\frac{1}{n^{6k-1}})$ by Algorithm 2 (see the proof of Theorem 1 *a)*), and with a probability of $\Omega(\frac{1}{n^{3k}})$ by Algorithm 3 (see Lemma 6 of [40], or the proof of Theorem 1 *b)*) in any of the N many independent draws in iteration t , if $\mathbf{X}_{t-1}^{[1]}$ is intersection-free and not optimal. As a result, we obtain by a similar proof as above Theorem 4 below.

Theorem 4. *Consider a TSP instance with n vertices located on an $m \times m$ grid such that no three of them are collinear. Assume that $|V - V^b| = k \in O(1)$, that we apply the max-min calibration 6 with $\pi_{\max} = 1 - \frac{1}{n}$, $\pi_{\min} = \frac{1}{n(n-2)}$, and set $\rho = 1, M = 1$, for some constant $\epsilon > 0$. Then:*

- a) If we set $N = \Omega(n^3 \cdot m^\epsilon)$, then with an overwhelming probability of $1 - e^{-\Omega(N/n^3)}$, Algorithm 1 finds an optimal solution within at most $n \cdot m^5 + n^{6k-4}$ iterations with the vertex-based random solution generation;
- b) If we set $N = \Omega(n^2 \cdot m^\epsilon)$, then with an overwhelming probability of $1 - e^{-\Omega(N/n^2)}$, Algorithm 1 finds an optimal solution within at most $n \cdot m^5 + n^{3k-2}$ iterations with the edge-based random solution generation.

Proof of Theorem 4. We only prove a). b) can be derived by a very similar argument. We define two random events as following:

\mathcal{E}_1 : for each $t \leq n \cdot m^5 + n^{6k-4}$, $f(\mathbf{X}_{t-1}^{[1]}) \geq f(\mathbf{X}_t^{[1]})$;

\mathcal{E}_2 : for each $t \leq n \cdot m^5 + n^{6k-4}$, if $\mathbf{X}_{t-1}^{[1]}$ is not intersection-free, then a 2-opt move happens in iteration t .

By a similar argument as the one for Theorem 3, we obtain that

$$\mathbf{P}[\mathcal{E}_1 \cap \mathcal{E}_2] \geq 1 - e^{-\Omega(N/n^3)}.$$

Let η be a random variable denoting the number of iterations for which $\mathbf{X}_{t-1}^{[1]}$ is intersection-free. Notice that, conditioned on $\mathcal{E}_1 \cap \mathcal{E}_2$, $\eta \geq n \cdot m^5$ implies that an optimal solution occurs within $n \cdot m^5 + n^{6k-4}$ iterations.

Conditioned on $\mathcal{E}_1 \cap \mathcal{E}_2$ and $\eta < n \cdot m^5$, there are at least $\Omega(n^{6k-4})$ iterations in which $\mathbf{X}_{t-1}^{[1]}$ is intersection-free, since each $\mathbf{X}_{t-1}^{[1]}$ is either intersection-free or not intersection-free. Note also that in each iteration in which $\mathbf{X}_{t-1}^{[1]}$ is intersection-free and not optimal, a $3k$ -opt move that turns $\mathbf{X}_{t-1}^{[1]}$ into an optimal solution happens with probability of at least

$$1 - (1 - \Omega(\frac{1}{n^{6k-1}}))^N.$$

This means for any fixed $t \in \mathbb{N}$, if $\mathbf{X}_{t-1}^{[1]}$ is intersection-free, then the probability of the event that $\mathbf{X}_t^{[1]}$ is optimal is bounded from below by

$$1 - (1 - \Omega(\frac{1}{n^{6k-1}}))^N.$$

Therefore, for any fixed $\Omega(n^{6k-4})$ many iterations in which the iteration-best solution $\mathbf{X}_{t-1}^{[1]}$ is intersection-free and not optimal, the probability of the event

that the corresponding $\Omega(n^{6k-4})$ many $\mathbf{X}_t^{[1]}$'s are still not optimal, is bounded from above by

$$(1 - \Omega(\frac{1}{n^{6k-1}}))^{N \cdot n^{6k-4}} = e^{-\Omega(N/n^3)}.$$

This means that, conditioned on $\mathcal{E}_1 \cap \mathcal{E}_2$ and $\eta < n \cdot m^5$, an optimal solution occurs within $n \cdot m^5 + n^{6k-4}$ iterations with a probability of $1 - e^{-\Omega(N/n^3)}$.

As a result, an optimal solution occurs within the first $n \cdot m^5 + n^{6k-4}$ iterations with a probability of $1 - e^{-\Omega(N/n^3)}$. \square

Theorem 4 shows a stochastic runtime of $n^3 m^{5+\epsilon} + n^{6k-1} m^\epsilon$ for Algorithm 1 equipped with the vertex-based solution generation, and a stochastic runtime of $n^3 m^{5+\epsilon} + n^{3k} m^\epsilon$ for Algorithm 1 equipped with edge-based solution generation, in the case of that $k = O(1)$. This is much better than the expected runtime

$$O(\mu \cdot n^3 m^5 + nm^5 + \mu \cdot n^{4k} (2k-1)!)$$

for $(\mu + \lambda)$ EA with sequential 2-opt mutations reported by Sutton et al [28]. However, we are not able to analyze the stochastic runtime in the case that $k = \omega(1)$, since $k = \omega(1)$ many interior points may require super-polynomially many iterations to turn an intersection-free solution into an optimal solution when a polynomial sample size is used.

4.3. An extension to *MMAS*

Although the stochastic runtimes in this article are obtained for iteration-best reinforcement, they can directly be extended to the case of best-so-far reinforcement. It is not difficult to see that all the Theorems also hold, when we use the best solution found so far instead of the iteration-best solution in the learning of \mathbf{W}_t in (4).

In the case of best-so-far reinforcement, the sample size N in Theorem 2 can even be further reduced to $\Omega(1)$ and the stochastic runtime stays the same. This is because then we do not need to employ a large N to guarantee that the next iteration does not become worse.

5. Conclusion

We have analyzed the stochastic runtime of a CE algorithm on two classes of TSP instances under two different random solution generation methods.

The stochastic runtimes are comparable with corresponding expected runtimes reported in the literature.

Our results show that the edge-based random solution generation method makes the algorithm more efficient for TSP instances in most cases. Moreover, moderately adapting the sample size to the problem size is helpful for efficiently finding an optimal solution with iteration-best reinforcement. For simple instances, a small sample size is enough, but for more difficult ones, one may need to use a relatively large sample size.

Although we considered only iteration-best reinforcement, our results also carry over to the best-so-far reinforcement. Our stochastic runtimes are better than the expected runtimes of the $(\mu + \lambda)$ EA on the grid instances. The EA randomly changes local structures of some of its current solutions by a Poisson distributed number of consecutive 2-exchange moves in every iteration, while our algorithm refrains from local operations on current solutions and only refreshes solutions by sampling from an evolving distribution. The solution reproducing mechanism in the EA stays the same throughout the optimization, only the current solutions in every iteration vary. However, the solution reproducing mechanism (sampling distribution) of our algorithm also evolves. This is the essential difference of MBS with traditional EAs. The comparison of our results with the expected runtimes in [28] therefore show that using a self-adaptive dynamic solution reproducing mechanism is helpful (in efficiently finding an optimal solution) when the search space becomes rugged. The stochastic runtimes in Theorem 4 are only valid for instances with a bounded number of interior points. In the future, it should be interesting to analyze the case that $|V - V^b| = \omega(1)$. This might also give more insight to the problem of \mathcal{RP} v.s. \mathcal{P} [49].

Our analysis is actually a kind of worst-case analysis. In this analysis, we are rather pessimistic. We analyze the optimization progress by only checking some very particular random events. This may not only underestimate the probability, but also overestimate the required number of iterations to reach an optimal solution. In the future, it should be of great interest to consider a smoothed runtime analysis over an ϵ -neighborhood of the n nodes in the real plane as has been done for the Simplex method by Spielman and Teng in their famous paper [50].

References

- [1] R. Y. Rubinstein, D. P. Kroese, The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning, Springer Science & Business Media, 2004.
- [2] R. Y. Rubinstein, Optimization of computer simulation models with rare events, *European Journal of Operational Research* 99 (1) (1997) 89–112.
- [3] R. Y. Rubinstein, The cross-entropy method for combinatorial and continuous optimization, *Methodology and computing in applied probability* 1 (2) (1999) 127–190.
- [4] M. Dorigo, T. Stützle, Ant colony optimization, Cambridge, Massachusetts: A Bradford Book, MIT Press, 2004.
- [5] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, *Swarm and Evolutionary Computation* 1 (3) (2011) 111–128.
- [6] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: A critical survey, *Annals of Operations Research* 131 (1-4) (2004) 373–395.
- [7] D. Whitley, A genetic algorithm tutorial, *Statistics and computing* 4 (2) (1994) 65–85.
- [8] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, Springer, 2003.
- [9] Z. Wu, Model-based heuristics for combinatorial optimization: a mathematical study of their asymptotic behavior, Ph.D. thesis, Institut für Angewandte Stochastik und Operations Research (IASOR), Technical University of Clausthal (2015).
- [10] T. Stützle, H. H. Hoos, MAX-MIN ant system, *Journal of Future Generation Computer Systems* 16 (2000) 889–914.
- [11] S. Droste, T. Jansen, I. Wegener, On the analysis of the (1+1) evolutionary algorithm, *Theoretical Computer Science* 276 (1-2) (2002) 51–81.

- [12] J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms, *Artificial Intelligence* 127 (1) (2001) 57–85.
- [13] F. Neumann, C. Witt, Runtime analysis of a simple ant colony optimization algorithm, Tech. rep., Department of Computer Science, University of Dortmund, Germany (2006).
- [14] C. Witt, Runtime analysis of the $(\mu + 1)$ ea on simple pseudo-boolean functions, *Evolutionary Computation* 14 (1) (2006) 65–86.
- [15] F. Neumann, C. Witt, Runtime analysis of a simple ant colony optimization algorithm, *Algorithmica* 54 (2) (2009) 243–255.
- [16] B. Doerr, F. Neumann, D. Sudholt, C. Witt, Runtime analysis of the 1-ant ant colony optimizer, *Theoretical Computer Science* 412 (17) (2011) 1629–1644.
- [17] W. J. Gutjahr, G. Sebastiani, Runtime analysis of ant colony optimization with best-so-far reinforcement, *Methodology & Computing in Applied Probability* 10 (3) (2008) 409–433.
- [18] Y. Zhou, J. He, A runtime analysis of evolutionary algorithms for constrained optimization problems, *IEEE Transactions on Evolutionary Computation* 11 (5) (2007) 608–619.
- [19] Y. Zhou, Runtime analysis of an ant colony optimization algorithm for tsp instances, *Evolutionary Computation IEEE Transactions on* 13 (5) (2009) 1083–1092.
- [20] P. S. Oliveto, C. Witt, Improved time complexity analysis of the simple genetic algorithm, *Theoretical Computer Science* 605 (15) (2015) 21–41.
- [21] D. Sudholt, C. Thyssen, Runtime analysis of ant colony optimization for shortest path problems, *Journal of Discrete Algorithms* 10 (10) (2012) 165–180.
- [22] A. Lissovoi, C. Witt, Runtime analysis of ant colony optimization on dynamic shortest path problems, *Theoretical Computer Science* 561 (2015) 73–85.

- [23] Y. Chen, X. Zou, Runtime analysis of a multi-objective evolutionary algorithm for obtaining finite approximations of pareto fronts, *Information Sciences* 262 (2014) 62–77.
- [24] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* 1 (1) (1997) 67–82.
- [25] F. Neumann, D. Sudholt, C. Witt, Analysis of different mmas aco algorithms on unimodal functions and plateaus, *Swarm Intelligence* 3 (2009) 35–68.
- [26] F. Neumann, I. Wegener, Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, *Theoretical Computer Science* 378 (2007) 32–40.
- [27] J. Reichel, M. Skutella, Evolutionary algorithms and matroid optimization problems, *Algorithmica* 57 (1) (2010) 187–206.
- [28] A. M. Sutton, F. Neumann, S. Nallaperuma, Parameterized runtime analyses of evolutionary algorithms for the planar euclidean traveling salesperson problem, *Evolutionary Computation* 22 (4) (2014) 595–628.
- [29] A. M. Sutton, J. Day, F. Neumann, A parameterized runtime analysis of evolutionary algorithms for max-2-sat, in: *Conference on Genetic & Evolutionary Computation*, 2012, pp. 433–440.
- [30] Y. Zhou, X. Lai, K. Li, Approximation and parameterized runtime analysis of evolutionary algorithms for the maximum cut problem., *IEEE Transactions on Cybernetics* 45 (8) (2015) 1491–1498.
- [31] Z. Wu, M. Kolonko, Asymptotic properties of a generalized cross entropy optimization algorithm, *IEEE Transactions on Evolutionary Computation* 18 (5) (2014) 658 – 673.
- [32] Z. Wu, M. Kolonko, R. H. Möhring, Stochastic runtime analysis of the cross entropy algorithm, under revision.
- [33] M. Held, R. M. Karp, A dynamic programming approach to sequencing problems, *Journal of the Society for Industrial and Applied Mathematics* 10 (1) (1962) 196–210.

- [34] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Tech. rep., Graduate School of Industrial Administration, CMU (1976).
- [35] M. T. Goodrich, R. Tamassia, Algorithm Design and Applications, Wiley, 2015.
- [36] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, *Journal of the ACM* 45 (5) (1988) 753–782.
- [37] J. S. B. Mitchell, A constant-factor approximation algorithm for tsp with pairwise-disjoint connected neighborhoods in the plane, in: *Twenty-Sixth Symposium on Computational Geometry*, 2010, pp. 183–191.
- [38] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Operations Research* 21 (2) (1973) 498–516.
- [39] P. T. D. Boer, D. P. Kroese, S. Mannor, R. Y. Rubinstein, A tutorial on the cross-entropy method, *Annals of Operations Research* 134 (1) (2005) 19–67.
- [40] T. Kötzing, F. Neumann, H. Röglin, C. Witt, Theoretical analysis of two aco approaches for the traveling salesman problem, *Swarm Intelligence* 6 (1) (2012) 1–21.
- [41] A. M. Sutton, F. Neumann, A parameterized runtime analysis of evolutionary algorithms for the euclidean traveling salesperson problem, in: *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI’12)*, AAAI press, 2012, pp. 1105–1111.
- [42] A. Costa, O. D. Jones, D. Kroese, Convergence properties of the cross-entropy method for discrete optimization, *Operations Research Letters* 35 (5) (2007) 573–580.
- [43] Z. Wu, M. Kolonko, Absorption in model-based search algorithms for combinatorial optimization, in: *Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 2014*, pp. 1744–1751.
- [44] M. Thomas, Machine learning, New Delhi: McGraw Hill Education India, 1997.

- [45] H. Asoh, H. Mühlenbein, On the mean convergence time of evolutionary algorithms without selection and mutation, in: *Parallel Problem Solving from Nature—PPSN III*, Springer, 1994, pp. 88–97.
- [46] M. Pirlot, General local search methods, *European Journal of Operational Research* 92 (3) (1996) 493–511.
- [47] M. A. Khamsi, W. A. Kirk, *An introduction to metric spaces and fixed point theory*, John Wiley,, 2001.
- [48] L. V. Quintas, F. Supnick, On some properties of shortest Hamiltonian circuits, *American Mathematical Monthly* 72 (9) (1965) 977–980.
- [49] W. Gasarch, Classifying problems into complexity classes, *Advances in Computers* 95 (2015) 239–292.
- [50] D. A. Spielman, S. H. Teng, Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time, *Journal of the Acm* 51 (3) (2004) 385–463.